

Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

1. (Currently amended) A processor, the processor implemented as a three way super scaler, pipelined architecture, the processor comprising:

an out-of-order microinstruction pointer ( $\mu$ IP) stack for storing pointers in a microcode ( $\mu$ code) execution core, the pointers placed on the out-of order microinstruction pointer stack and removed from the microinstruction pointer stack before it is known if a sequence of microinstructions pointed to by the pointers is valid.

2. (Previously presented) The processor of claim 1 in which entries in the  $\mu$ IP stack comprise:

an entry number field;  
a microinstruction pointer ( $\mu$ IP) field;  
a back pointer field;  
a retirement indicator field; and  
a return pointer field.

3. (Original) The processor of claim 2 in which the  $\mu$ IP field is 14-bits wide.

4. (Original) The processor of claim 3 in which the  $\mu$ IP field has a microinstruction pointer ( $\mu$ IP) pushed by a first microoperation ( $\mu$ Op) code and used by a second  $\mu$ Op code.

5. (Original) The processor of claim 2 in which the back pointer field has a pointer to a next entry in the  $\mu$ IP stack for a micro-type of service ( $\mu$ TOS) bit to point to after a  $\mu$ Op.

6. (Original) The processor of claim 2 in which the retirement indicator field has an indication of whether an entry has retired.

7. (Original) The processor of claim 2 in the return pointer field a pointer to a location in a retirement stack to which an entry is copied after being retired.

8. (Currently amended) A method executed in a processor, the processor implemented as a three way super scaler, pipelined architecture, the method comprising:

executing microcode ( $\mu$ code) addressed by pointers stored in an out-of-order microinstruction pointer ( $\mu$ IP) stack, the pointers placed on the out-of order microinstruction pointer stack and removed from the microinstruction pointer stack before it is known if a sequence of microinstructions pointed to by the pointers is valid; and  
manipulating the  $\mu$ IP stack with a set of microinstructions.

9. (Previously presented) The method of claim 8 in which entries in the stack have an entry number field, a microinstruction pointer ( $\mu$ IP) field, a back pointer field, a retirement indicator field and a return pointer field.

10. (Original) The method of claim 9 in which the  $\mu$ IP pointer field is 14-bits wide.

11. (Original) The method of claim 10 in which the  $\mu$ IP pointer field has a microinstruction pointer ( $\mu$ IP) pushed by a first microoperation ( $\mu$ Op) code and used by a second  $\mu$ Op code.

12. (Original) The method of claim 9 in which the back pointer field has a pointer to a next entry in the  $\mu$ IP stack for a micro-type of service ( $\mu$ TOS) bit to point to after a  $\mu$ Op.

13. (Original) The method of claim 9 in which the retirement indicator field has an indication of whether an entry has retired.
14. (Original) The method of claim 9 in which the return pointer field contains a pointer to a location in a retirement stack to which an entry is copied after being retired.
15. (Original) The method of claim 9 in which manipulating comprises:
  - pushing a next  $\mu$ IP on to the  $\mu$ IP stack; and
  - using the next  $\mu$ IP in an intermediate field as a target  $\mu$ IP in a jump operation.
16. (Original) The method of claim 9 in which manipulating comprises:
  - taking a value of an intermediate field of a microoperation ( $\mu$ Op); and
  - pushing the value on to the  $\mu$ IP stack.
17. (Original) The method of claim 9 in which manipulating comprises:
  - popping a value off the  $\mu$ IP stack; and
  - replacing a current  $\mu$ Op intermediate field.
18. (Original) The method of claim 9 in which manipulating comprises:
  - popping a value off of the  $\mu$ IP stack; and
  - jumping to that value.
19. (Original) The method of claim 9 in which manipulating comprises:
  - reading a value off the  $\mu$ IP stack; and
  - replacing a  $\mu$ Op's intermediate field with the value.
20. (Original) The method of claim 9 in which manipulating comprises setting the  $\mu$ IP stack pointers to reset.

21. (Original) The method of claim 9 further comprising providing a set of pointers that point to different entries in the  $\mu$ IP stack.

22. (Original) The method of claim 21 in which the set of pointers includes a  $\mu$ TOS pointer that points to a top of the  $\mu$ IP stack.

23. (Original) The method of claim 21 in which the set of pointers includes a  $\mu$ Alloc pointer that points to a next allocated entry in the  $\mu$ IP stack.

24. (Original) The method of claim 21 in which the set of pointers includes a NextRet pointer that points to a next entry in the  $\mu$ IP stack to be deallocated.

25. (Original) The method of claim 21 in which the set of pointers includes  $\mu$ RetTos pointer that points at a retired top of the  $\mu$ IP stack.

26. (Original) The method of claim 8 in which the  $\mu$ OPs include an `ms_call`  $\mu$ OP that takes a next  $\mu$ IP, pushes the next  $\mu$ IP on the  $\mu$ IP stack, and uses the next  $\mu$ IP in an intermediate field as a target  $\mu$ IP of a jump.

27. (Original) The method of claim 8 in which the  $\mu$ OPs include an `ms_push`  $\mu$ OP that takes a value in an intermediate field and pushes the value on the  $\mu$ IP stack.

28. (Original) The method of claim 8 in which the  $\mu$ OPs include an `ms_pop`  $\mu$ OP that pops a value off the  $\mu$ IP stack and replaces the value with the  $\mu$ OP's intermediate field.

29. (Original) The method of claim 8 in which the  $\mu$ OPs include an `ms_return`  $\mu$ OP that pops a value off of the  $\mu$ IP stack and jumps to that  $\mu$ IP.

30. (Original) The method of claim 8 in which the  $\mu$ OPs include an `ms_tos_read`  $\mu$ OP that reads a value off the  $\mu$ IP stack and replaces this  $\mu$ OP's intermediate field.

31. (Original) The method of claim 8 in which the  $\mu$ OPs include an `ms_muip_stack_clear`  $\mu$ OP that sets the  $\mu$ IP stack pointers to reset.

32. (Currently amended) A computer program product residing on a computer readable medium having instructions stored thereon which, when executed by the processor, cause the processor to:

execute microcode ( $\mu$ code) addressed through pointers stored in an out-of-order microinstruction pointer ( $\mu$ IP) stack, the pointers placed on the out-of order microinstruction pointer stack and removed from the microinstruction pointer stack before it is known if a sequence of microinstructions pointed to by the pointers is valid; and  
manipulate the  $\mu$ IP stack with a set of microinstructions.

33. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

push a next  $\mu$ IP on to the  $\mu$ IP stack; and  
use the next  $\mu$ IP in an intermediate field as a target  $\mu$ IP in a jump operation.

34. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

take a value of an intermediate field of a microoperation ( $\mu$ Op); and  
push the value on to the  $\mu$ IP stack.

35. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

pop a value off the  $\mu$ IP stack; and  
replace a current  $\mu$ Op intermediate field with the value.

36. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

pop a value off of the  $\mu$ IP stack; and  
jump to that value.

37. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

read a value off the  $\mu$ IP stack; and  
replace a  $\mu$ Op's intermediate field with the value.

38. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

set the  $\mu$ IP stack pointers to reset.